Proceedings of the Federated Conference on Computer Science and Information Systems pp. 145–153
DOI: 10.15439/2018F248
ISSN 2300-5963 ACSIS, Vol. 15

Combining the Syntactic and Semantic Representations of Mizar Proofs

Karol Pąk
University of Białystok,
Ciołkowskiego 1M, 15-245 Białystok, Poland
Email: pakkarol@uwb.edu.pl

***Abstract*—The Mizar system provides two representations of the proofs present in its library. The syntactic representation preserves the human-friendly rich Mizar language, where the meaning of structures and expressions is still influenced by their context. The semantic one, on the other hand, explicitly reflects the meaning of all elements present in the proof scripts, however many features of the Mizar language are eliminated.**

In this article, we overcome the limitations of both representations of proofs, by proposing a method combining them. We show that we can simultaneously maintain the richness of the language and provide access to the derived proof information. We discuss how such combined information closer corresponds to that present in other proof assistant languages, for example that of Isabelle/Isar.

I. Introduction

ONE of the most recognizable features of the Mizar system [1] is its highly human-oriented proof environment. For over four decades the Mizar project has developed an environment that allows to create formal reasoning similar to how it is done in informal mathematical practice. Many actions at various levels have been taken to ensure this environment:

- Linguistically motivated dependent soft type system that closely reflects how most of the mathematicians use mathematical objects and how they categorize them.
- Rich expression language that provides complex symbol overloading and notational constructs, e.g., the meaning of individual symbols is strongly influenced by the context, as in textbooks.
- The Mizar natural-deduction proof that try to approximate the way how informal proofs are written, where the deduction is not steered by an explicate list of rules that indicates how to match a statement to its proof.

These possibilities are reflected in the size of one of the largest formal libraries, the Mizar Mathematical Library (MML) [2] that includes many domains that have not been formalized elsewhere. However, the Mizar system is the only tool that can fully operate on the content of the library written in such rich language.

a) Related works: Obviously, there have been a number of attempts to explore the content of the MML by external tools. Most of them ensure access to the semantic representation of MML where many proof details are easily accessible for a human reader, starting with easy semantic searching tools as MML Query [3], variants of XML format [4], MMT logical framework [5] or provide the extensive theorems database of MPTP [6] that can be served as a tool in automated theorem proving and machine learning [7].

The Mizar system also provides two ways of access to the syntactic representation. These are *Weakly Strict Mizar* (WSX) [8] and *More Strict Mizar* (MSX) [9], where MSX is an extension of WSX. However, both human-friendly syntactic representations are still too far from semantic one and selecting access, we have to choose between these two.

A lot of work has been done to cross-verify MML by external tools that struggle with many problems such as the Mizar logic that goes a little bit beyond the first-order logic and the Mizar type system. Pioneering and the largest translation of MML to the TPTP untyped first-order language has been done by Urban [10], where higher-order problems have been verified with an extension of TPTP language [11]. Kunčar [12] has attempted to translate the MML as transparent higher-order logic theories, however, his concept was not able to cover more advanced features of the Mizar type system. Successful attempt to extract and cross-verify higher-order problems using higher-order automated theorem provers Satallax and LEO-II have been done by C. Brown [13]. However, these translations are fixed from the point of view of a further development of the MML, i.e., they do not allow any modification or further development.

Urban's MizAR system [14] facilitates the search for a given justification using machine learning and automated reasoning. It returns a list of premises needed for the current goal, which in most cases can be directly used in the script, as in the case of justification generated by the Sledgehammer subsystem for Isabelle/HOL [15]. However, in some cases to use obtained list of premises, authors need to carefully modify the

The paper has been supported by the resources of the Polish National Science Center granted by decision n°DEC-2015/19/D/ST6/01473.

Mizar proof script `environ` and/or create manually a few auxiliary steps.

b) Isabelle/Mizar: is a project whose goals are to: (1) specify the Mizar foundations fully formally in the Isabelle logical framework [16] and (2) cross-verify all the proofs in MML in resulting fully specified logic. The work on these can in the longer term lead to (3) the creation of a Mizar like environment in the Isabelle logical framework, which would provide Mizar foundations and its various mechanisms and allow users to further develop proofs in typed set theory using various Isabelle mechanisms.

There has been a lot of progress on the first goal. The environment [17], [18] has an equivalent of the Mizar dependent type system including Mizar-like structures [19], as well as higher-order concepts, such as set comprehensions and schemes [20]. We have recently developed an automated translation of the statements; however, a large gap between the syntactic and semantic Mizar representations significantly hinders work on an automatic export of the MML proofs. The proofs should be as close to the syntactic Mizar representation as possible, but at the same time, we need to use the semantics, e.g., to identify objects and reconstruct their lists of hidden arguments.

c) Contribution: This paper introduces a representation of Mizar proofs that combines the syntactic and semantic ones. Most of the original syntax is preserved, annotated by the information how particular constructs have been interpreted by Mizar, which allows the processing of the proofs by external systems. The particular contributions are:

- An elegant presentation of Mizar proofs using implicit types assigned to frequently used variable names. This combines the syntactic and semantic representation of the Mizar reserve mechanism.
- A simplification of the Mizar proof outlines to the fix–assume–show outlines present in most declarative proof languages, e.g., Isabelle/Isar [21]. This involves eliminating several constructions, such as the take steps that indicates a suitable term for instantiating an existentially quantified thesis. In particular the concept of the take step has no equivalent in most other proof languages [21].
- Lists of dedicated rules that determine the reasoning patterns for each sub-proof with the fix–assume–show proof outline present in most declarative proof languages. We propose Isabelle/Isar rules as an example.
- A human-friendly rendering of our Mizar representation.

The paper is constructed as follows. In Section II we discuss the Mizar proof concepts lost in the Mizar semantic exports. In Section III we briefly describe a method that matches syntactic and semantic representations. Then we present a reconstruction of these concepts based on the matched representations. In Section IV we present a way to simplify Mizar proofs to logical framework formats preserving reconstructed concepts. Finally in Section V we show an example formalization created in the Isabelle/Mizar environment that uses a rebuilt Mizar proof.

II. Mizar Semantic Export

The scope of this paper does not allow to fully explain the details in the Mizar system including the semantic representation (for such details see [22], [4]). We will point out the main problems and solutions using a single example that states a natural basic property of the set inclusion (Fig. 1). Additionally, the deduction used there as the property justification is quite similar to *informal* one and can be analyzed by a reader who does not have experience with the Mizar system.

a) Hidden Types and Quantifiers: According to the Mizar syntax each statement has first-order form where atomic predicative formulas are combined with classical logic connectives and quantifiers. Also note, that each quantifier has to associate a Mizar type (types correspond to first-order predicates) with all bounded variables. However, the Mizar Analyzer can not infer this type automatically based on the context of the variables as is the case for most type-theory based systems, since Mizar disambiguates the meaning of each symbol based on types of its arguments (corresponding to type inference in an intersection-type system). To avoid specifying *explicitly* the types of all bound variables at their quantifiers, the Mizar system provides a *reservation* mechanisms that allows global associating variable names with their types. For example, `object` that is the type of the variable `x` is not mentioned in the sentence ex `x` st in Fig. 1 but is imported from the reservation reserve `x,y` for `object`.

The type of a reserved variable can be skipped not only in the declaration of a quantifier, but also in each Mizar construction which requires it (compare lines 7 and 11). Moreover, for convenience, universal quantifiers that bind reserved variables can be implicit. Additionally, in many systems, free variables that correspond to *implicit* quantifiers in a given statement are automatically introduced to a sub-deduction that justifies the statement. Obviously such user support is welcome in the human-readable export, especially since the equivalent of the Mizar reserve mechanism has already been provided in Isabelle/Mizar. Unfortunately, Mizar reconstructs these quantifiers and corresponding introduction steps very early, between WSX and MSX representations and does not distinguish reconstructed objects in the XML semantic representation[1]. As a consequence, there is no difference in the semantic representation between the theorem presented in Fig. 1 and Fig. 2. On the other hand, we can easily distinguish this information in the WSX representation, but based on this representation we can not provide enough information

[1]The semantic representation of the theorem presented in Fig. 1 where hidden types and quantifiers are fully reconstructed is available in the HTMLization of the current Mizar library http://mizar.uwb.edu.pl/version/current/html/xboole_0.html#t8

```
1  reserve x,y for object,
2          X,Y for set;
3  theorem
4    X c< Y implies ex x st x in Y & X c= Y\{x}
5  proof
6    assume A1: X c< Y;                                  thesis: X c< Y implies ex x st x in Y & X c= Y\{x}
7    then consider x such that                                 thesis: ex x st x in Y & X c= Y\{x}
8    A2: x in Y and A3: not x in X by Def8,TARSKI:def 3;
9    take x;                                                 thesis: x in Y & X c= Y\{x}
10   thus x in Y by A2;                                            thesis: X c= Y\{x}
11   let y;                                                  thesis: y in X implies y in Y\{x}
12   assume A4: y in X;                                            thesis: y in Y\{x}
13   then y<>x by A3;
14   then A5: not y in {x} by TARSKI:def 1;
15   X c= Y by A1;
16   then y in Y by A4;                                                  thesis: verum
17   thus then thesis by Def5,A5;
18 end;
```

Fig. 1. An example Mizar style theorem, originally occurring as XBOOLE_0:8 (eighth theorem in the Mizar proof scripts XBOOLE_0), with *implicit* thesis explicitly shown. The theorem states that if X is a proper subset of Y (X c< Y), then there exists a member x of Y (x in Y) for which X is a subset of the complement of the singleton {x} in Y.

```
1  theorem
2    for X,Y being set st X c< Y holds
3      ex x being set st x in Y & X c= Y\{x}
4  proof
5    let X,Y be set;
6    assume A1: X c< Y;
```

Fig. 2. An example of the semantically equivalent formulation of the theorem presented in Fig. 1 together with a fragment of its justification.

to explore the MML. Note that information about disambiguating symbols and their hidden arguments are missing there and are very hard to reconstruct for any external tool.

b) Normal Form: The reservation system is one of the three main reasons why we are forced to combine information from syntactic and semantic representations. To simplify the grammar of the semantic representation, the Mizar Analyzer also transforms each formula to the Mizar normal form (MNF) which uses only selected logical connectives, such as $\neg$, a generalization of $\wedge$ for n-arguments, the universal quantifier $\forall$ and $\bot$. A lot of work has been done by Urban to minimize the consequences of normalization [4]. He built directly into the Mizar Analyzer a hint system that is visible as an additional attribute pid in selected nodes of the XML semantic representation. This system should allow the reconstruction of the original formula from the normalized one based on pid-s, e.g., every implication $\alpha \rightarrow \beta$ is replaced by the formula $\neg_{-4}(\wedge_{-5}(\alpha, \neg_{-6}(\beta)))$ where subscripts represent pid–values. However, these hints are often lost in the normalization process, since, e.g., the Analyzer eliminates double negation together with the corresponding pid-s. In consequence, there are several cases (a few percent of the library)[2], where the original formulation is different than the HTMLization generated with the pid-support even if we omit reconstructed hidden quantifiers.

It is important to note that equivalent reformulation of statements in a Mizar proof script does not affect its correctness, since most of Mizar verifier's modules are based on the MNF including the Reasoner which check the applicability of *Skeleton steps* – discussed in Section IV that operates on the thesis in a given proof. For comparison, equivalent reformulation of statements is not possible in most declarative proof languages, as all reasoning pattern must precisely correspond to the related statements and proofs.

c) More advanced pid*–problems:* The reconstruction of the original logical conjunctions and quantifiers was one of the additional tasks in the pid system created by Urban. His system mainly focused on solving the conflict between *patterns* and *constructors*, that we only sketch here (for more details see [10]). Note that every Mizar defined object (i.e, a function, a type, a predicate, an attribute or a structure) together with its list of arguments with their types (and a result type if applicable) and positions of each visible arguments constitutes the Mizar pattern. Mizar constructors are just absolute identifiers (in the environment of a given article) of Mizar objects of which the patterns are translated during their full identification by the Mizar Analyzer. Obviously, a given overloaded pattern can be translated into different constructors. Unfortunately, different patterns can be translated into the same constructor, since, e.g., synonyms and antonyms of predicates and adjectives inherit the constructor from their ancestor. To ensure full control over the many-to-many

[2] For example, the ZFMISC_1 article contains 139 theorems and the differences occur in 8 cases, i.e, theorems: 6, 19, 22, 37, 53, 58, 112, 138.

relationship, the number of constructor (nr) and pattern (pid) should be associated with each object. However, many pid-s are lost or arguments of patterns are incorrectly reconstructed. We can observe this in the HTMLization as *technical* constructors rather than the corresponding patterns or as missing values in the lists of arguments[3]. Such defects do not have a significant negative impact for the rendered HTML, but are unacceptable in every cross-verification of the MML.

III. Disambiguated Syntactic Mizar Export

The problems indicated in Section II are typical, if we want to obtain access to the MML based only on the semantic representation. Therefore we develop an application that combines the semantic and syntactic informations. Obviously there are many inconveniences in this approach, since the semantic representation is completely rewritten by the Mizar Analyzer with respect to the syntactic one and contains only the information that are absolutely necessary for the checking proof steps. We combine the information in three stages.

a) Top level: First, we match all the items from the two representations, where often a few semantic steps correspond to a single syntactic one. For example, a step that introduces variables X, Y in Fig. 2 (see 5th line) is hidden in Fig. 1, but for semantic comparison, both variables are introduced in independent steps that are followed by additional steps where the corresponding modified thesis is formulated.

b) Logic connectives and quantifiers: Next we match syntactic and semantic representation of each statement to find corresponding atomic formula. For this purpose we transform every syntactically represented statement imitating the Mizar Analyzer process such as the normalization and then we compare the obtained formula with the corresponding semantic representation of this statement. We transform also all formulas into a system of abstractions and applications in meta logic

```
<logic id=("ball"|"hidden_ball"|"bex"|"iff"|
  "impl"|"or"|"and"|"not"|"False"|"True")\>
```

that should be easy-to-read by external tools, since e.g., our system directly corresponds to logical framework application and abstraction. Note that the constant hidden_ball is semantically equivalent to ball but corresponds to a universal quantifier that is originally hidden. Additionally, we distinguish types of variables that are imported form *reservations*. For example, hidden quantifiers that bind variables with types imported from *reservations* in the statement of theorem presented in Fig. 1 obtained the following our representation:

[3] See for example http://mizar.uwb.edu.pl/version/current/html/pboole.html#CC4 where the expression includes V8 rather than non-empty, V9 rather than empty-yielding, and the argument A is missing in the type ManySortedSet of A.

```
<proposition label="xboole_0_th_8">
 <app>
  <logic id="hidden_ball" type="o" args="2"
     argsType="ty_abs"/>
  <ReservationType id="X">
   <const id="HIDDENM2" type="ty" args="0"
      argsType="set"/>
  </ReservationType>
  <abs id="X" type="set" args="0">
   <app>
    <logic id="hidden_ball" type="o" args="2"
       argsType="ty_abs"/>
    <ReservationType id="Y">
     <const id="HIDDENM2" type="ty" args="0"
        argsType="set"/>
    </ReservationType>
    <abs id="Y" type="set" args="0">
    ...
```

where the constant HIDDENM2 corresponds to the Mizar set that is the second type definition (called mode in Mizar) in article HIDDEN. Note that the name directly corresponds to the absolute constructor name proposed in [10] and the OMDoc node <OMS module="HIDDEN"name="M2"/> (according to the
naming scheme proposed in [5]).

c) Atomic propositions: Matching at the atomic proposition level is quite natural. Generally, we just match predicates and then recursively terms and subterms to disambiguate them. However, we have to take into account Mizar local abbreviations that are fully unfolded in the semantic representation. It is also important to note that most of the Mizar objects have different numbers and order of arguments in compared representations, since the semantic representation contains visible arguments of each Mizar object, but also their hidden arguments calculated by the Mizar Analyzer. We explore these differences to access hidden arguments at the syntactic level, and use them just like visible arguments. Additionally, as in the case of logic connectives, we present every object using an application of meta-constant that corresponds to unique pattern of the object and list of its arguments.

For example, let us consider the statement of theorem SUBSET_1:13 presented in Fig. 3. It states that the set difference of sets A and B is equal to the intersection of A and the complement of B in the given universal set E.

The universal set E does not appear *explicitly* in the statement, but is necessary to determine the complement set B`. Additionally, the difference (represented as \) and the intersection (/\) are originally defined for arbitrary sets, however the Mizar redefinitions (for more detail see [22]) change types of return values in these functors for Subset of E, if both arguments are also Subset of E. Therefore, E is a hidden argument of these three functors if we want to fully reflect the meaning of this statement. The representation of the fact is presented in Fig. 4.

To provide a human-friendly access to our representation we also build an initial system that automatically generate pdf files that visualize our representation as well as HTMLization of the MML

```
reserve E for set,
        A,B for Subset of E;
theorem :: SUBSET_1:13
  A \ B = A /\ B`;
```

Fig. 3. An example Mizar theorem whose statement contains hidden arguments.

```
<app>
 <const id="XBOOLE_0R4" type="o" args="2"
   argsType="set"/>
 <app>
  <const id="SUBSET_1K7" type="set" args="3"
    argsType="set"/>
  <var id="A" type="set" args="0" argsType="set"/>
  <var id="E" type="set" args="0" argsType="set"/>
  <var id="B" type="set" args="0" argsType="set"/>
 </app>
 <app>
  <const id="SUBSET_1K9" type="set" args="3"
     argsType="set"/>
  <var id="A" type="set" args="0" argsType="set"/>
  <var id="E" type="set" args="0" argsType="set"/>
  <app>
   <const id="SUBSET_1K3" type="set" args="2"
      argsType="set"/>
   <var id="B" type="set" args="0" argsType="set"/>
   <var id="E" type="set" args="0" argsType="set"/>
  </app>
 </app>
</app>
```

Fig. 4. The formula `A \ B = A /\ B`` represented in our format. These are necessary to decode the complete information. `SUBSET_1K7` corresponds to the pattern `_ \ _`, defined in the Mizar article `SUBSET_1` to represent the difference of sets in an universe, where the universe is a hidden argument and is calculated by Mizar from types of the sets. Similarly, `SUBSET_1K9` and `SUBSET_1K3` correspond to patterns `_ /\ _` and `_`', respectively.

visualize the semantic representation. We use a presentation inspired by that of Isabelle rendering of its formalizations, in particular applied to Isabelle/Mizar that combines selected components of Isabelle/Isar and Mizar[4]. In particular, the theorem presented in Fig. 3 is expressed as follows:

mtheorem subset_1_th_13:
∀E: ⟨ set HIDDENM2 ⟩ .
∀A: ⟨ Subset SUBSET_1M2 of E ⟩ .
∀B: ⟨ Subset SUBSET_1M2 of E ⟩ .
A \SUBSET_1K7⟨E⟩ B = XBOOLE_0R4
A /\SUBSET_1K9⟨E⟩ B `SUBSET_1K3⟨E⟩

where hidden quantifiers and types imported from reservations are highlighted as well as hidden arguments are visible in subscripts. Additionally, identifiers indicate absolute patterns and links indicate absolute constructors in the HTMLization of the current MML.

[4] Readers can check automatically generated pdf files (generated now for 104 initial Mizar articles) at the author's web site http://alioth.uwb.edu.pl/~pakkarol/fedcsis2018/, stylized for the Isabelle/Isar language.

```
reserve x for object, X,Y for set;
theorem
  X/\Y = X implies X c= Y
proof
  assume that A1: for x st x in X/\Y holds x in  X and
              A2: X c= X/\Y and
              A3: ex x st x in X & not x in Y;
```

Fig. 5. An example Mizar assumption where two predicates (equality and inclusion) are unfolded in one skeleton step.

IV. Simplification of Mizar proofs by cut introduction

The Mizar proof style, inspired by Jaśkowski [23], provides various natural deduction steps (called *Skeleton steps* in Mizar). The steps generally modify the current part of a given *thesis* that still remains to be proven. Thesis is the same as the current goal at the beginning of every proof, but further it becomes *implicit*, as is done in informal proofs. Indeed, mathematicians do not often indicate what has been done or what is left in the middle of proofs. It means that Mizar authors must know the current thesis and predict how it will be changed by a particular skeleton step to finish a given proof. Mizar proof is finished if the thesis is reduced to `verum` (*true*). Then the Mizar Reasoner tries to adapt the skeleton steps proposed by authors even if this requires unfolding the definitions of several predicates.

A. Unfolded Predicates

An example of an *implicit* unfolded definition is presented in Fig. 1 in line 11. The generalization step (keyword let) can be used if the current thesis is a universally quantified formula, but the current thesis after line 10 is a formula (`X c= Y\{x}`) that becomes a universally quantified formula if we unfold a definition of set inclusion. Such an approach gives a lot of freedom for authors, but is very hard to control by existing external tools. It is important to note that the Mizar semantic representation has been enriched by Urban with a list of definitions that the Mizar Reasoner actually needs to unfold in every step. Such information is sufficient to cross-verify a given thesis modification done by Reasoner (for more detail see, [24]). However this information is not sufficient to determine reasoning pattern, since it does not determine positions of unfolded predicates, or even their number. An example of an assumption accepted by Reasoner supported by two definitions, namely the definition of equality as two inclusions and the definition of inclusion is presented in Fig. 5. The equality `X/\Y = X` is introduced as two inclusions, where the first one `X/\Y c= X` has been unfolded, and further the indirect proof is started where the inclusion in the indirect assumption not `X c= Y` has been unfolded. Determination of a reasoning pattern for such an assumption is a severe problem. However, combining the syntactic and semantic information we can automatically

eliminate the more advanced skeleton steps, generating the corresponding reasoning patterns while introducing fewest changes in the reasoning.

B. Procedure Overview

Note that we can transform modified thesis by a given reasoning step back to the thesis before this step or to an equivalent formula, if we take into account the meaning of the definitions unfolded there (for more details see [24]). It means that we can reconstruct an equivalent of the thesis by analyzing the skeleton steps from the end of a given proof, if there are only *simple* (i.e., without definitional expansions) kinds of steps, such as generalizations, assumptions, conclusions (or shorter let–assume–thus) that correspond directly to the Isar fix–assume–show. Moreover, in such cases we can indicate a list of natural deduction rules that precisely correspond to the related created thesis and proof. In our representation we only use implication introduction and the following four rules (expressed in the Isabelle syntax):

lemma impMI: (A1 ⟹ A2 ⟶ C) ⟹
  A1 ∧ A2 ⟶ C
lemma conjMI: C2 ⟹ C1 ⟹ C1 ∧ C2
lemma ballI: (⋀x. x be D ⟹ P(x)) ⟹
  inhabited(D) ⟹ ∀x:D. P(x)
lemma bexI: P(x) ⟹ x be D ⟹
  inhabited(D) ⟹ ∃x:D. P(x)

where impMI connects uncurry and impI; conjMI is a modification of conjI; ballI, bexI are bounded quantifier introduction and elimination rules which apart from the condition ensure that the given Mizar types are inhabited. These correctly correspond to the Mizar foundations (see [17]). Note that conjMI corresponds to the Mizar conclusion where a given proposition is a conjunct of the current thesis and impI separates a list of conjunctions in an assumption to give them independent labels as follows:

have A ∧ B ∧ C
proof(rule conjMI,rule conjMI)
  show A ⟨proof⟩
  show B ⟨proof⟩
  show C ⟨proof⟩
qed
have A ∧ B ∧ C ⟶ D
proof(rule impMI,rule impMI,rule impI)
  assume a: A **and** b: B **and** c: C
  show D ⟨proof⟩
qed

As shown in Fig. 5, a reconstructed thesis can not be easily matched to a given thesis in a proof, if some definitions have been unfolded in a let–assume–thus step. Therefore, in our approach we introduce a cut in the reasoning at every place where such steps occur. Let us fix such a let–assume–thus step. We encapsulate a part of deduction beginning from the step using the created list of rules as a sub-deduction that proves the reconstructed thesis (the correctness condition of such cut introduction have been developed in [25]). Then we replace this step by a conclusion where the original thesis is given as the proposition and we refer to the sub-deduction and unfolded definitions and as the justification. An example of such a cut introduced to our representation is presented in Fig. 6.

```
have ∀ y : ⟨ object HIDDENM1 ⟩ .
    y in HIDDENR3 X →
     y in HIDDENR3 Y \ XBOOLE_0K4 { TARSKIK1 x }
proof(rule ballI,rule impI)
 fix y being ⟨ object HIDDENM1 ⟩
 assume A4: y in HIDDENR3 X
 hence y <> HIDDENR2 x using A3;
 hence A5: ¬ y in HIDDENR3 { TARSKIK1 x }
    using tarski_def_1;
 have X c= TARSKIR1 Y using A1;
 hence y in HIDDENR3 Y using A4;
 thus y in HIDDENR3 Y \ XBOOLE_0K4 { TARSKIK1 x }
    using xboole_0_def_5, A5;
qed
thus X c= TARSKIR1 Y \ XBOOLE_0K4 { TARSKIK1 x }
   using tarski_def_3;
```

Fig. 6. An example of a cut introduction related to the skeleton step located in line 11 in Fig. 1.

C. take *steps*

The Mizar take is a kind of skeleton step that is a challenge for other declarative proof languages, including expressing the proofs in Isabelle/Mizar, as such steps cannot be omitted. take indicates terms suitable for instantiating an existentially quantified thesis. Such terms can be constructed using any available constants in Mizar. For comparison, there is a limitation for kinds of constants in the Isabelle/Isar language i.e., constants introduced inside obtain steps have to be available before a deduction where we use them to construct such suitable term. Unfortunately, the obtain step is the only equivalent of the Mizar consider (for more detail see [21]) and most of take steps are using consider constants. A cut introduction is one and only one solution that we introduce in our representation. For example, we can introduce the following cut

```
show ∃ t : object HIDDENM1 .
 t in HIDDENR3 Y ∧
   X c= TARSKIR1 Y \ XBOOLE_0K4 { TARSKIK1 t }
proof(rule bexI[of _ x],rule conjMI)
 show x in HIDDENR3 Y using A2;
 have ∀ y : ⟨ object HIDDENM1 ⟩ .
    y in HIDDENR3 X →
    y in HIDDENR3 Y \ XBOOLE_0K4 { TARSKIK1 x }
 proof(rule ballI,rule impI)...
 thus X c= TARSKIR1 Y \ XBOOLE_0K4 { TARSKIK1 x }
    using tarski_def_3;
qed
```

in relation to the step take x; presented in Fig. 1.

Note that to introduce such cut we have to extract a given term and also its type, but the type can be *implicit* even in the semantic representation. Generally, we can extract this type comparing the thesis before and after a given take step, but not in most cases where some definitions have been unfolded. For them we use the following solution. Let us regard t as such take step and

denote by `f` the first skeleton step after `t` that is not a take step where we can not extract a type. First, we encapsulate a part of deduction beginning from `f` as a sub-deduction that proves a thesis that corresponds to `f`. Then we formulate a conclusion with the original thesis of `t` and as a justification we refer to the sub-deduction, unfolded definitions in `t`, but also in all skeleton steps between `t` and `f`; and a list of bexI rules substituted by terms given in `t` and skeleton steps between `t` and `f`. In the case of the take step presented in Fig. 1. proposed solution should introduce the following cut:

```
show ∃ t : object HIDDENM1 .
 t in HIDDENR3 Y ∧
   X c= TARSKIR1 Y \ XBOOLE_0K4 { TARSKIK1 t }
proof-
 have x in HIDDENR3 Y ∧
    X c= TARSKIR1 Y \ XBOOLE_0K4 { TARSKIK1 x }
 proof(rule conjMI)...
 thus ?thesis using unfolded definitions bexI[of _ x];
qed
```

D. The rest of skeleton steps

The remaining Mizar skeleton steps not explained so far are: given, hereby and also per cases steps that play a similar role as skeleton steps, but do not modify a considered thesis.

The given step is an abbreviation for an assume step that as a valid proposition introduces an existential statement and a consider step that creates a fresh constant and provides access to the instantiated existential statement with the constant. We replace every such step via assume, consider/obtain.

To describe the hereby step, we must first introduce the now concept, since hereby is a simply abbreviation of thus now. In the Mizar language now opens a sub-deduction where the proved statement is not written explicitly but is reconstructed from the sub-deduction. The sub-deduction can be conducted with the support of all kinds of skeleton steps with only one restriction that each take steps have to be formulated as "take *new constant=term*" to indicate all terms that should be replaced by a variable bounded by the appropriate existential quantifier. Weaker equivalent of the now blocks are present in some proof languages, for example the Isabelle/Isar `{...}` concept supports only deduction via fix–assume where the last `have` step is chosen as the conclusion. Therefore we replace the now blocks by *normal* steps with reconstructed statements in our export.

The per cases step is a kind of step that generally reflects the idea of the informal proof by cases, where a thesis has to be justified under each of logically complementary alternatives, but not necessarily with identical skeleton steps. We encapsulate the sub-deduction in each case as a justification of the corresponding implication "*case assumption* implies *reconstructed thesis*".

V. Our representation as a next stage to cross-verify MML in Isabelle

In this section we describe possibilities of our representation in relation to the needs of the Isabelle logical framework and in particular Isar reasoning patterns. In our previous work [18], we defined a unique and faithful equivalent of the Mizar dependent type system and higher-order concepts as an Isabelle object logic. This equivalent has been tested so far only on a manually reformalized part of the MML. However, the experience gained during manual re-formalization showed directions in which we can bring the Isar language closer to the Mizar one as well as unattainable goals, e.g., the take step. Our manual attempts to generate Isar reasoning patterns also showed that with enough effort, we can indicate corresponding lists of rules even for very intricate deductions. However, such list are too sensitive to minor changes, even in *simple* deductions. Our representation of proofs is an attempt to solve these problems on the MML side.

Opportunities offered by this representation at the moment have been visualized on a re-formalization of the theorem presented in Fig. 1 created in our Isabelle environment. The re-formalization is presented in Fig. 7 and reflects all elements contained in the automatically generated visualization[5].

The example demonstrates the usefulness of the reconstructed reserve concept (Section III) that is welcome in the human-readable export. Note that the reserve command is our Isabelle/Mizar equivalent of the Mizar reserve mechanism that collects variable names with their types. Therefore, we do not need mention the types of `X`, `Y`, `x` in quantified formulas same as in the Mizar proof scripts. Additionally, we can hide the first two quantifiers in the statement of the theorem, since the mtheorem command automatically binds free reserve variables, introduces them into the sub-deduction, and adds corresponding propositions of the shape "*term* is *type*" to the *background knowledge* of the proof stored by a designated theorem list (`ty`). Then the knowledge, extended by some additional informations is added to the list of premises by the proof method mauto before auto call (more detail in our formalization).

We can also observe consequences of the introduced cuts and the simplicity of the generated reasoning pattern described in Section IV. Note that Isabelle/Isar does not accept a given sub-deduction as the justification of a particular statement, even if it accepts justifications for all the steps in a sub-deduction, since the given reasoning pattern does not precisely correspond to the statement and the sub-deduction.

VI. Conclusion

We have introduced a combination of the Mizar syntactic and semantic proof representations and presented

[5]See http://alioth.uwb.edu.pl/~pakkarol/fedcsis2018/mispdf/xboole_0.pdf

```
reserve X,Y for set
reserve x for object
mtheorem xboole_0_th_8:
 ∀ X. ∀ Y. X c< Y ⟶
   (∃ x. x in Y ∧ X c= Y\{x} )
proof -
 have ∀ X. ∀ Y. X c< Y ⟶
   (∃ t:object. t in Y ∧ X c= Y\{t} )
 proof(rule balll,rule balll,rule impl)
  fix X assume [ty]:X be set
  fix Y assume [ty]:Y be set
  assume A1: X c= Y
  then obtain x where [ty]: x be object and
   A2: x in Y and A3: ¬ x in X using xboole_0_th_6
   by mauto
  show ∃ t : object. t in Y ∧ X c= Y\{t}
  proof(rule bexI[of _ x],rule conjMI)
   show x in Y using A2 by auto
   have ∀ y : object. y in X ⟶ y in Y\{x}
   proof(rule balll,rule impl)
    fix y assume [ty]:y be object
    assume A4: y in X
    hence y <> x using A3 by auto
    hence A5: ¬ y in {x} using tarski_def_1
     by auto
    have X c= Y using A1 xboole_0_def_8
     by mauto
    hence y in Y using A4 tarski_def_3 by mauto
    thus y in  Y\{x} using xboole_0_def_5 A5
     by mauto
   qed mauto
   thus X c= Y\{x} using tarski_def_3 by mauto
  qed mauto
 qed mauto
 thus ?thesis by mauto
qed
```

Fig. 7. An example Isabelle/Mizar reasoning that exactly corresponds to the combined representation of the proof script presented in Fig. 1. Note that the highlighted part of reasoning can be removed from the script without influence for its correctness just like in the Mizar proof scripts (for more detail see our formalization)

a number of possibilities that such combined data offers. We rebuild the Mizar natural deduction style to the fix–assume–thus proof outlines present in declarative proof modes, including that of Isabelle/Isar. The transformation preserves all Mizar components using cut introduction. This eliminates all the Mizar natural deduction constructions for which adequate equivalent constructs do not exist in other systems. In particular, it reduces the distance between the Mizar and Isabelle/Isar proof styles, which we showed in an experiment in which a transformed MML proof could be directly cross-verified in Isabelle/Mizar. The original and transformed proofs for the first 50 Mizar articles are available at:

http://alioth.uwb.edu.pl/~pakkarol/fedcsis2018/

Future work could target a further reduction of the distance between Mizar and Isabelle. The MML export combining the syntactic and semantic representations, as well as the Isabelle/Mizar object logic and its packages are under active development. Many Mizar concepts are still not completely expressed in Isabelle (e.g., the Mizar reconsider construction) or they have not been sufficiently verified (e.g., the Mizar structures [22] which are used directly or indirectly in the latter 74% of the MML). We believe that under a suitable proof translation ATPs are strong enough to accept all justifications accepted by the Mizar checker. However, to make the automation more efficient and practical, it might be necessary to extract additional knowledge from the semantic representation used as an initial information by the checker.

References

[1] G. Bancerek, C. Byliński, A. Grabowski, A. Korniłowicz, R. Matuszewski, A. Naumowicz, K. Pąk, and J. Urban, "Mizar: State-of-the-art and Beyond," in *Intelligent Computer Mathematics - International Conference, CICM 2015*, ser. LNCS, M. Kerber, J. Carette, C. Kaliszyk, F. Rabe, and V. Sorge, Eds., vol. 9150. Springer, 2015. doi: 10.1007/978-3-319-20615-8_17 pp. 261–279.

[2] G. Bancerek, C. Byliński, A. Grabowski, A. Korniłowicz, R. Matuszewski, A. Naumowicz, and K. Pąk, "The role of the Mizar Mathematical Library for interactive proof development in Mizar," *J. Autom. Reasoning*, 2017. doi: 10.1007/s10817-017-9440-6. [Online]. Available: https://doi.org/10.1007/s10817-017-9440-6

[3] G. Bancerek and P. Rudnicki, "Information retrieval in MML," in *Mathematical Knowledge Management, MKM 2003*, ser. LNCS, A. Asperti, B. Buchberger, and J. H. Davenport, Eds., vol. 2594. Springer, 2003. doi: 10.1007/3-540-36469-2_10. ISBN 3-540-00568-4 pp. 119–132. [Online]. Available: https://doi.org/10.1007/3-540-36469-2_10

[4] J. Urban, "XML-izing Mizar: Making semantic processing and presentation of MML easy," in *Mathematical Knowledge Management (MKM 2005)*, ser. LNCS, M. Kohlhase, Ed., vol. 3863. Springer, 2005. ISBN 3-540-31430-X pp. 346–360.

[5] M. Iancu, M. Kohlhase, F. Rabe, and J. Urban, "The Mizar Mathematical Library in OMDoc: Translation and applications," *J. Autom. Reasoning*, vol. 50, no. 2, pp. 191–202, 2013. doi: 10.1007/s10817-012-9271-4

[6] J. Urban, "MPTP 0.2: Design, implementation, and initial experiments," *J. Autom. Reasoning*, vol. 37, no. 1–2, pp. 21–43, 2006. doi: 10.1007/s10817-006-9032-3

[7] C. Kaliszyk and J. Urban, "MizAR 40 for Mizar 40," *J. Autom. Reasoning*, vol. 55, no. 3, pp. 245–256, 2015. doi: 10.1007/s10817-015-9330-8

[8] A. Naumowicz and R. Piliszek, "Accessing the Mizar library with a weakly strict Mizar parser," in *Intelligent Computer Mathematics, CICM 2016*, ser. LNCS, M. Kohlhase, M. Johansson, B. R. Miller, L. de Moura, and F. W. Tompa, Eds., vol. 9791. Springer, 2016. doi: 10.1007/978-3-319-42547-4_6 pp. 77–82. [Online]. Available: http://doi.org/10.1007/978-3-319-42547-4_6

[9] C. Byliński and J. Alama, "New Developments in Parsing Mizar," in *Intelligent Computer Mathematics - 11th International Conference, AISC 2012, 19th Symposium, Calculemus 2012, 5th International Workshop, DML 2012, 11th International Conference, MKM 2012, Systems and Projects, Held as Part of CICM 2012*, ser. 7362, J. Jeuring, J. A. Campbell, J. Carette, G. D. Reis, P. Sojka, M. Wenzel, and V. Sorge, Eds., vol. 5170. Springer, 2012. doi: 10.1007/978-3-642-31374-5 pp. 427–431.

[10] J. Urban, "Translating Mizar for first order theorem provers," in *Mathematical Knowledge Management, Second International Conference, MKM 2003, Bertinoro, Italy, February 16-18, 2003, Proceedings*, ser. LNCS, A. Asperti, B. Buchberger, and J. H. Davenport, Eds., vol. 2594. Springer, 2003. doi: 10.1007/3-540-36469-2_16 pp. 203–215.

[11] J. Urban and G. Sutcliffe, "ATP-based cross-verification of Mizar proofs: Method, systems, and first experiments," *Math.*

in Computer Science, vol. 2, no. 2, pp. 231–251, 2008. doi: 10.1007/s11786-008-0053-7

[12] O. Kunčar, "Reconstruction of the Mizar type system in the HOL Light system," in *WDS Proceedings of Contributed Papers: Part I – Mathematics and Computer Sciences*, J. Pavlu and J. Safrankova, Eds. Matfyzpress, 2010, pp. 7–12.

[13] C. E. Brown and J. Urban, "Extracting higher-order goals from the Mizar Mathematical Library," in *Intelligent Computer Mathematics (CICM 2016)*, ser. LNCS, M. Kohlhase, M. Johansson, B. R. Miller, L. de Moura, and F. W. Tompa, Eds., vol. 9791. Springer, 2016. doi: 10.1007/978-3-319-42547-4_8 pp. 99–114.

[14] J. Urban, P. Rudnicki, and G. Sutcliffe, "ATP and presentation service for Mizar formalizations," *J. Autom. Reasoning*, vol. 50, no. 2, pp. 229–241, 2013. doi: 10.1007/s10817-012-9269-y. [Online]. Available: https://doi.org/10.1007/s10817-012-9269-y

[15] J. C. Blanchette, D. Greenaway, C. Kaliszyk, D. Kühlwein, and J. Urban, "A learning-based fact selector for Isabelle/HOL," *J. Autom. Reasoning*, vol. 57, no. 3, pp. 219–244, 2016. doi: 10.1007/s10817-016-9362-8. [Online]. Available: http://dx.doi.org/10.1007/s10817-016-9362-8

[16] M. Wenzel, L. C. Paulson, and T. Nipkow, "The Isabelle framework," in *Theorem Proving in Higher Order Logics, 21st International Conference, TPHOLs 2008*, ser. LNCS, O. A. Mohamed, C. A. Muñoz, and S. Tahar, Eds., vol. 5170. Springer, 2008. doi: 10.1007/978-3-540-71067-7_7 pp. 33–38.

[17] C. Kaliszyk, K. Pąk, and J. Urban, "Towards a Mizar environment for Isabelle: Foundations and language," in *Proc. 5th Conference on Certified Programs and Proofs (CPP 2016)*, J. Avigad and A. Chlipala, Eds. ACM, 2016. doi: 10.1145/2854065.2854070 pp. 58–65.

[18] C. Kaliszyk and K. Pąk, "Semantics of Mizar as an Isabelle object logic," *J. Autom. Reasoning* 2018. doi: doi.org/10.1007/s10817-018-9479-z. [Online]. Available: https://doi.org/10.1007/s10817-018-9479-z

[19] ——, "Progress in the independent certification of Mizar Mathematical Library in Isabelle," in *Proceedings of the 2017 Federated Conference on Computer Science and Information Systems, FedCSIS 2017*, M. Ganzha, L. A. Maciaszek, and M. Paprzycki, Eds., 2017. doi: 10.15439/2017F289 pp. 227–236.

[20] ——, "Isabelle formalization of set theoretic structures and set comprehensions," in *Mathematical Aspects of Computer and Information Sciences, MACIS 2017*, ser. LNCS, J. Blamer, T. Kutsia, and D. Simos, Eds., vol. 10693. Springer, 2017.

[21] M. Wenzel and F. Wiedijk, "A comparison of Mizar and Isar," *J. Autom. Reasoning*, vol. 29, no. 3-4, pp. 389–411, 2002. doi: 10.1023/A:1021935419355

[22] A. Grabowski, A. Korniłowicz, and A. Naumowicz, "Mizar in a nutshell," *J. Formalized Reasoning*, vol. 3, no. 2, pp. 153–245, 2010. doi: 10.6092/issn.1972-5787/1980

[23] S. Jaśkowski, "On the rules of suppositions," *Studia Logica*, vol. 1, 1934.

[24] J. Urban and G. Sutcliffe, "ATP cross-verification of the Mizar MPTP challenge problems," in *Logic for Programming, Artificial Intelligence, and Reasoning, 14th International Conference, LPAR 2007, Yerevan, Armenia, October 15–19, 2007, Proceedings*, 2007. doi: 10.1007/978-3-540-75560-9_39 pp. 546–560.

[25] K. Pąk, "Methods of lemma extraction in natural deduction proofs," *J. Autom. Reasoning*, vol. 50, no. 2, pp. 217–228, February 2013. doi: 10.1007/s10817-012-9267-0. [Online]. Available: http://dx.doi.org/10.1007/s10817-012-9267-0